Algebraic Block Red-Black Ordering Method for Parallelized ICCG Solver With Fast Convergence and Low Communication Costs

Takeshi Iwashita, *Member, IEEE,* and Masaaki Shimasaki, *Member, IEEE*

***Abstract*—This paper proposes a new parallelized incomplete Cholesky conjugate gradient (ICCG) solver effective on a small-scale multiprocessor system. The new method is based on a new reordering technique, namely the block red-black ordering method. Its parallel performance is evaluated in a finite edge-element eddy-current analysis. A numerical test shows that the proposed method is effective on a small number of processors due to fast convergence and low communication costs.**

***Index Terms*—Algebraic block red-black ordering, black-box type solver, incomplete Cholesky conjugate gradient (ICCG) method, parallel processing.**

I. Introduction

THE PRESENT paper proposes a new technique for parallel processing the incomplete Cholesky conjugate gradient (ICCG) method [1] on parallel computers using a small number of processors (2–8 processors), for example, SMP PCs or small-scale PC clusters. An effective use of these small-scale parallel computation environments has recently become significant in fast electromagnetic (EM) field analyses for the practical design of EM machines.

While the ICCG method is the most popular solver for symmetric positive-definite linear systems arising in EM field computations, parallel processing the ICCG method is difficult due to the forward–backward substitutions included in the iteration kernels. Several strategies have been presented for the parallelization of the ICCG method [2], [3]. In the IEEE COMPUMAG-Evian 2001, the authors reported a black-box type parallelized ICCG solver, which is called the algebraic multicolor ordering method [4]. The previous method, which is oriented to high-class parallel supercomputers, achieves both very high parallelism and fast convergence. But, if the number of used processors is limited, another method that attains better solver performance must be developed. The strategy for obtaining better performance is based on improvement of convergence and reduction of communication (synchronization) costs. Based on this strategy, the authors have proposed the block red-black ordering method in finite difference analyses [5]. In this method, the entire set of grid points is divided into several or many blocks, and red-black ordering is applied to the blocks. An increase in the block-size results in an improvement of convergence. The advantages of this method are: 1) fewer synchronization points (only one in parallelized substitution) and 2) an easy set of optimal block-size for achieving the best convergence. This paper proposes an application of this new ordering to a black-box type solver for unstructured finite-element analyses, that is, the algebraic block red-black ordering method.

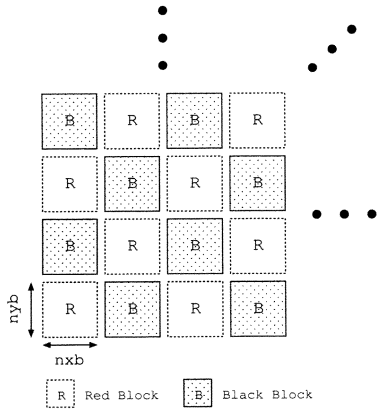

Fig. 1. Partitions of blocks in block red-black ordering.

II. Algebraic Block Red-Black Ordering

This paper solves a linear system of equations having a positive or semi-positive definite symmetric coefficient matrix by means of the ICCG method. Parallel processing forward and backward substitutions is mainly discussed here, because other kernels of the ICCG method can be easily parallelized [2].

A. Block Red-Black Ordering

This subsection explains block red-black ordering [5] that is a parallel processing technique of the ICCG method proposed in the context of structured finite difference analysis. In this technique, all grid-points are first divided into several blocks and red-black ordering is applied to the blocks. Fig. 1 shows partitions of blocks based on block red-black ordering in a two-dimensional (2-D) case. Each block contains $nxb \times nyb$ nodes. Since blocks with the identical color have no data dependency, computations regarding the blocks can be parallelized in each color. Fig. 2 illustrates a parallelized procedure of the substitutions by using the block red-black ordering. The method has an

Manuscript received June 18, 2002.

T. Iwashita is with the Academic Center for Computing and Media Studies, Kyoto University, Kyoto 606-8501 Japan (e-mail: iwashita@media.kyoto-u.ac.jp).

M. Shimasaki is with the Department of Electrical Engineering, Kyoto University, Kyoto Japan (e-mail: simasaki@kuee.kyoto-u.ac.jp).

Digital Object Identifier 10.1109/TMAG.2003.810531

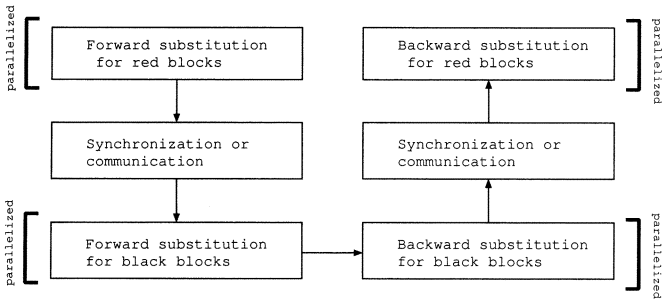

Fig. 2. Parallelized substitutions in block red-black ordering method.

advantage in that only one synchronization (or communication) point exists in one substitution. From the viewpoint of convergence, a larger block-size results in a better convergence. This property is explained by using ordering graph theory [5], [6]. Based on this property, an optimal block-size is easily set when the number of used processors is given.

B. Algebraic Block Red-Black Ordering

This paper develops the block red-black ordering method into a black-box type method, which is called the algebraic block red-black ordering method. The new method algebraically performs the division of unknowns into groups (blocks) and the assignment of colors (red or black). This is a key procedure of the algebraic block red-black ordering and is described in detail.

The block-size bs is first given. The method of determining an optimal block-size is discussed in the following subsection. The procedure for determining the color and the block number of unknowns is given as follows.

- Step 1: First bs unknowns are assigned in *Red-Block 1*.
- Step 2: All the unknowns having data relationships with *Red-Block 1* are assigned in *Black-Block 1*. In this step and all following steps, the data relationships between the unknowns are found by examining positions of nonzero entries in the coefficient matrix $\boldsymbol{A}$. If the (p_i, p_j)-entry of $\boldsymbol{A}$ is nonzero, the p_ith unknown has a data relationship with the p_jth unknown.
- Step 3: If the number of unknowns in *Black-Block 1* is less than bs, an appropriate number of unknowns are added to the block.
- Step 4: All the unknowns related to the *Black-Block 1* are assigned in *Red-Block 2*. An appropriate number of unknowns are also added to the block if the number of unknowns in the block does not reach bs.
- Above procedures from Step 2 to Step 4 are recursively performed until the color and the block number of all the unknowns are determined.

Fig. 3 summarizes an assignment procedure for the ith red and black blocks. This setup procedure guarantees the independence between blocks in each color, which is required for parallel processing. After the assignment of the unknowns to the blocks, the unknowns are reordered from the red blocks to the black blocks. The coefficient matrix and the vectors are also renumbered. The computational costs for the setup and the renumbering process are usually negligible in terms of the total computation time [7].

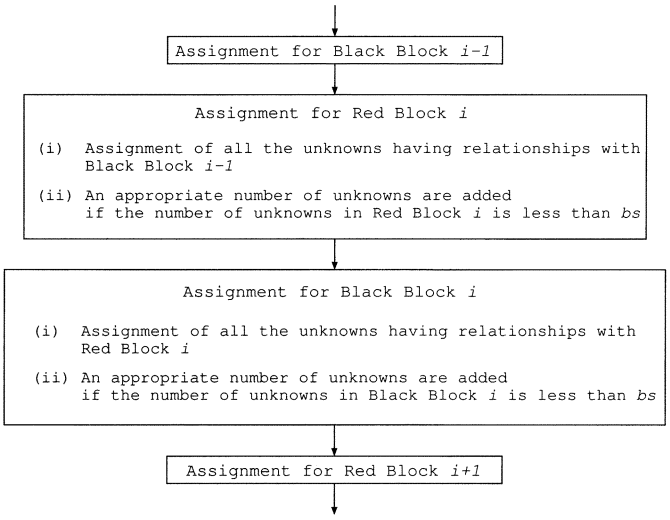

Fig. 3. Allocation of the ith red block and the ith black block.

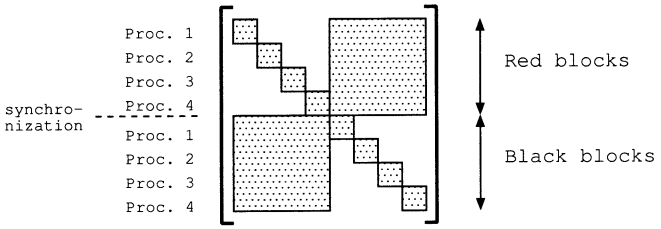

Fig. 4. Renumbered coefficient matrix (four processors, $Nr = Nb = 4$).

C. Parallelized ICCG Method Based on Algebraic Block Red-Black Ordering

This subsection describes the procedure of parallelized forward and backward substitutions in the algebraic block red-black ordering method. Since floating operations in the substitutions does not depend on a choice of ordering, the total computational cost for the algebraic block red-black ordering method is the same as that for the sequential ICCG method.

Let $\tilde{\boldsymbol{A}}\tilde{\boldsymbol{x}} = \tilde{\boldsymbol{b}}$ be the renumbered linear system obtained from the preceding subsection. Fig. 4 illustrates the renumbered coefficient matrix $\tilde{\boldsymbol{A}}$. When the solution vector $\tilde{\boldsymbol{x}}$ is split corresponding to the blocks as

$$\tilde{\boldsymbol{x}} = (\,\tilde{\boldsymbol{x}}_R \quad \tilde{\boldsymbol{x}}_B\,) = \left(\,\tilde{\boldsymbol{x}}_{r_1} \;\vdots\; \tilde{\boldsymbol{x}}_{r_{Nr}} \quad \tilde{\boldsymbol{x}}_{b_1} \;\vdots\; \tilde{\boldsymbol{x}}_{b_{Nb}}\right) \tag{1}$$

the coefficient matrix is also written as

$$\tilde{\boldsymbol{A}} = \left(\begin{array}{ccc|ccc} \tilde{\boldsymbol{A}}_{r_1,r_1} & & \boldsymbol{O} & & \tilde{\boldsymbol{A}}_{r_1,B} & \\ & \ddots & & & \vdots & \\ \boldsymbol{O} & & \tilde{\boldsymbol{A}}_{r_{Nr},r_{Nr}} & & \tilde{\boldsymbol{A}}_{r_{Nr},B} & \\ \hline & \tilde{\boldsymbol{A}}_{b_1,R} & & \tilde{\boldsymbol{A}}_{b_1,b_1} & & \boldsymbol{O} \\ & \vdots & & & \ddots & \\ & \tilde{\boldsymbol{A}}_{b_{Nb},R} & & \boldsymbol{O} & & \tilde{\boldsymbol{A}}_{b_{Nb},b_{Nb}} \end{array}\right) \tag{2}$$

where Nr is the number of red blocks, Nb is the number of black blocks, and the subscripts R, B, r_i, and b_i denote the unknowns of all red blocks, all black blocks, the ith red block,

and the ith black block, respectively. In (2), $\tilde{\boldsymbol{A}}_{\alpha,\beta}$ represents the relationship between $\tilde{\boldsymbol{x}}_\alpha$ and $\tilde{\boldsymbol{x}}_\beta$. The incomplete factorized matrix of $\tilde{\boldsymbol{A}}$ for preconditioning is given by

$$\tilde{\boldsymbol{L}} = \left(\begin{array}{ccc|ccc} \tilde{\boldsymbol{L}}_{r_1,r_1} & & \boldsymbol{O} & & & \\ & \ddots & & & \boldsymbol{O} & \\ \boldsymbol{O} & & \tilde{\boldsymbol{L}}_{r_{Nr},r_{Nr}} & & & \\ \hline & \tilde{\boldsymbol{L}}_{b_1,R} & & \tilde{\boldsymbol{L}}_{b_1,b_1} & & \boldsymbol{O} \\ & \vdots & & & \ddots & \\ & \tilde{\boldsymbol{L}}_{b_{Nb},R} & & \boldsymbol{O} & & \tilde{\boldsymbol{L}}_{b_{Nb},b_{Nb}} \end{array}\right) \tag{3}$$

where

$$\tilde{\boldsymbol{A}} \simeq \tilde{\boldsymbol{L}}\tilde{\boldsymbol{L}}^T. \tag{4}$$

From (3), the forward substitution $\tilde{\boldsymbol{L}}\tilde{\boldsymbol{y}} = \tilde{\boldsymbol{r}}$ is given by the following steps. For the unknowns in the ith red block, the substitution is given by

$$\tilde{\boldsymbol{y}}_{r_i} = \tilde{\boldsymbol{L}}_{r_i}^{-1}\tilde{\boldsymbol{r}}_{r_i}. \tag{5}$$

Each processor simultaneously performs (5) for one or several red blocks, which are assigned to the processor. After the results of the forward substitution for the red blocks $\tilde{\boldsymbol{y}}_R$ are communicated among the processors, the forward substitution for the ith black block is given by

$$\tilde{\boldsymbol{y}}_{b_i} = \tilde{\boldsymbol{L}}_{b_i}^{-1}\left(\tilde{\boldsymbol{r}}_{b_i} - \tilde{\boldsymbol{L}}_{b_i,R}\tilde{\boldsymbol{y}}_R\right). \tag{6}$$

Since the processors share the value of $\tilde{\boldsymbol{y}}_R$, each processor computes (6) for the black blocks assigned to the processor in parallel. The backward substitution is also parallelized in the same way. Accordingly, Fig. 2 also illustrates the procedure of the parallelized substitution based on the *algebraic* block red-black ordering method. In one parallelized substitution, only one communication (synchronization) point exists. If the numbers of nonzero entries are seriously different among blocks, load balance can be improved by adjusting the number of additional unknowns in Steps 3 and 4 in Section II-B. The degree of parallelism obtained from the present method is given by the number of blocks in one color, that is, Nr and Nb. Since the proposed method is based on a concept of block red-black ordering, a larger block-size generally results in a better convergence rate. Therefore, a block-size as large as possible for the number of processors is an optimal block-size. In this case, the degree of parallelism of the method is equal to the number of processors, and each processor carries out computations regarding one red block and one black block. Thus, the optimal number of block-size $bs_{\rm opt}$ is given by

$$bs_{\rm opt} = n/2N_p \tag{7}$$

where n and N_p are the total number of unknowns and the number of processors, respectively.

III. Results

A. Test Model and Computation Environment

In this analysis, we use the IEEJ standard benchmark model of three-dimensional (3-D) eddy current analyses [8] for evaluation of the proposed method. The analyzed model is discretized by first-order brick-type edge elements. Table I lists the discretization data. The EM field equations are solved by using the Galerkin method with the A-formulation and the backward time difference method. A gauge condition $\phi = 0$ is imposed in the conductive region, where ϕ is the electric scalar potential. No gauge condition is imposed in the nonconductive region, which leads to a semi-positive definite coefficient matrix. Therefore, the shifted coefficient matrix, which is positive-definite, is used in the incomplete factorization [9]. The shift parameter that is called the acceleration factor is chosen to be 1.07. The original linear system is assembled with natural ordering.

TABLE I
Discretization Data

Number of volume elements	327680
Number of nodes	342225
Number of unknowns	1011920
Time step	1 msec

The present analyses were implemented on two kinds of parallel computers. First, one is a distributed memory parallel supercomputer Fujitsu VPP-800, and second one is a SMP-type (shared memory) parallel computer Fujitsu GP-7000F model 900 which is a multiprocessor system composed of SUN SPARC-compatible processors. The VPP-800 has high-speed data transfer units for communication among processor elements, and its peak performance is 8 Gflops per processor element. On the other hand, the GP-7000F does not have a special hardware support for communications, and its peak performance is 1.2 Gflops per processor. The parallelized program code is written by using FORTRAN and MPI. The convergence criterion of the ICCG method is given by $||\boldsymbol{r}||_2/||\boldsymbol{b}||_2 = ||\tilde{\boldsymbol{r}}||_2/||\tilde{\boldsymbol{b}}||_2 < 10^{-7}$, where $\boldsymbol{r}$ and $\boldsymbol{b}$ are the residual vector and the right-hand side vector of the original linear system, respectively. The parallel performance of the proposed method is evaluated in the first time step.

B. Parallel Performance

In order to evaluate the proposed method, we compare the new method with the block ICCG method [7] and the algebraic multicolor ordering method that was proposed in [4] (COMPUMAG-Evian). In the following discussion, we use the notations BICCG, AMC, and ABRB to represent the block ICCG method, the algebraic multicolor ordering method, and the algebraic block red-black ordering method, respectively.

1) Numerical Tests on VPP-800: Table II shows the elapsed time, the number of iterations, and the speed-up ratio compared with sequential implementation of the ICCG method in numerical tests on the VPP-800 (Supercomputer). Both of our methods, the AMC and the ABRB show a better solver performance than the BICCG. In the BICCG, some sets of entries in the coefficient matrix are ignored for parallel processing in the incomplete factorization, which results in an increase of iterations due to a decline in the preconditioning effect [7].

Next, we compare the ABRB with the AMC. In this numerical test, the ABRB attains better solver performance on two or four processors. This is due mainly to a high convergence rate and reduction in communication costs. When the number of processors increases, the optimal block-size in the ABRB is also increased, which results in a decline in convergence. Accordingly, the AMC obtains a better solver performance in

TABLE II
NUMERICAL RESULTS ON VPP-800 (SUPER COMPUTER). (a) ICCG ON IP (NATURAL ORDERING). (b) BLOCK ICCG METHOD. (c) ALGEBRAIC MULTICOLOR ORDERING METHOD. (d) ALGEBRAIC RED–BLACK ORDERING METHOD

N_p	Elapsed time (sec)	# of iterations
1	2119	366

(a)

N_p	Elapsed time (sec)	# of iterations	Speed-up
2	1394	530	1.52
4	1028	779	2.06
8	550.5	834	3.85

(b)

N_p	# of color	Elapsed time (sec)	# of iterations	Speed-up
2	60	1194	390	1.77
4	60	606.3	390	3.49
8	60	308.4	390	6.87

(c)

N_p	Elapsed time (sec)	# of iterations	Speed-up
2	1070	374	1.98
4	552.2	383	3.84
8	312.5	424	6.78

(d)

the case of 8 processors. But, the elapsed time of the ABRB is almost equal to that of the AMC. Moreover, while the above result of the AMC is based on the optimal number of colors (60 colors) obtained in many numerical tests with various numbers of colors [4], the ABRB attains its good solver performance with the block-size set automatically. This is important in a case in which many different models are solved in the design of devices or machines.

2) Numerical Tests on GP-7000F: Table III lists the results of numerical tests performed on the GP-7000F (SMP). First we compare the ABRB with the AMC. Since the GP-7000F does not have a high-speed data transfer unit, the ABRB attains a better parallel performance than the AMC in the case of any number of processors. In particular, when two or four processors are used, the advantage of the ABRB is large. This is because the AMC has many communication (synchronization) points in parallelized substitutions and requires great communication cost.

Next, a comparison of the ABRB and the BICCG is discussed. Since the BICCG has no communication points in parallelized substitutions, the BICCG is effective in a parallel computation environment having low-grade network facilities among processors. On the other hand, the ABRB has the advantage of a high convergence rate while the BICCG suffers from a large decline in convergence. Consequently, the ABRB shows a better solver performance in the case of two or four processors, and the BICCG is advantageous in an eight-processor case. But, the ABRB attains almost the same solver performance as does the BICCG even if eight processors are used. Moreover, since hardware and software for communication among processors are rapidly developing, the ABRB will become more advantageous in the future.

IV. CONCLUSION

This paper proposes a new parallelized ICCG method effective on a small-scale multiprocessor system, which is called the algebraic block red-black ordering method. The new method is a black-box type solver that is proposed for application to a linear system of equations with a random sparse coefficient matrix. The method has only one communication (synchronization) point in each substitution. A high convergence rate is achieved by using an optimal block-size that is easily set. Accordingly, the new method attains a high parallel performance especially in a small-scale parallel computation environment.

TABLE III
NUMERICAL RESULTS ON GP-7000F (SMP). (a) ICCG ON IP (NATURAL ORDERING). (b) BLOCK ICCG METHOD. (c) ALGEBRAIC MULTICOLOR ORDERING METHOD. (d) ALGEBRAIC RED–BLACK ORDERING METHOD

N_p	Elapsed time (sec)	# of iterations
1	3704	368

(a)

N_p	Elapsed time (sec)	# of iterations	Speed-up
2	2507	534	1.48
4	1888	790	1.96
8	1031	843	3.59

(b)

N_p	# of color	Elapsed time (sec)	# of iterations	Speed-up
2	40	3415	464	1.08
4	40	2084	464	1.78
8	40	1160	464	3.19
2	60	2905	390	1.28
4	60	1546	390	2.40
8	60	1172	390	3.16
2	80	3883	533	0.95
4	80	2070	533	1.79
8	80	1190	533	3.11

(c)

N_p	Elapsed time (sec)	# of iterations	Speed-up
2	2337	378	1.58
4	1296	385	2.86
8	1067	430	3.47

(d)

REFERENCES

[1] J. Meijerink and H. A. van der Vorst, "An iterative solution method for linear systems of which the coefficient matrix is a symmetric M-matrix," *Math. Comput.*, vol. 31, pp. 148–162, 1977.

[2] R. Barrett *et al.*, *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods*. Philadelphia, PA: SIAM Press, 1994.

[3] I. S. Duff and H. A. van der Vorst, "Developments and trend in the parallel solution of linear systems," *Parallel Comput.*, vol. 25, pp. 1931–1970, 1999.

[4] T. Iwashita and M. Shimasaki, "Algebraic multicolor ordering for parallelized ICCG solver in finite element analyses," *IEEE Trans. Magn.*, vol. 38, pp. 429–432, Mar. 2002.

[5] ——, "Block red-black ordering method for parallel processing of ICCG solver," *Springer Lecture Notes in Computer Science*, vol. 2327, pp. 175–189, 2002.

[6] S. Doi and A. Lichnewsky, "A graph-theory approach for analyzing the effects of ordering on ILU preconditioning," INRIA Rep. 1452, 1991.

[7] T. Iwashita and M. Shimasaki, "Parallel processing of 3-D eddy current analysis with moving conductor using parallelized ICCG solver with renumbering process," *IEEE Trans. Magn.*, vol. 36, pp. 1504–1509, July 2000.

[8] T. Nakata, N. Takahashi, T. Imai, and K. Muramatsu, "Comparison of various methods of analysis and finite elements in 3-D magnetic field analysis," *IEEE Trans. Magn.*, vol. 27, pp. 4073–4076, Sept. 1991.

[9] K. Fujiwara, T. Nakata, and H. Fusayasu, "Acceleration of convergence characteristic of the ICCG method," *IEEE Trans. Magn.*, vol. 29, pp. 1958–1961, Sept. 1993.